

REMARKS

This application has been carefully reviewed in light of the Office Action dated July 28, 2008. Claims 1-10 remain in this application. Claims 1 and 2 are the independent Claims. Claims 1 and 9 have been amended. Claim 10 is a new claim. It is believed that no new matter is involved in the amendments or arguments presented herein.

Reconsideration and entrance of the amendment in the application are respectfully requested.

Claim Objection

Claim 1 was objected to because of an informality. In response, applicant has amended claim 1 removing "compress code type informations." It is believed that the amendment addresses the concern of above objection. Reconsideration and withdrawal of the above objection are respectfully requested.

Art-Based Rejections

Claims 1-4, and 7-9 were rejected under 35 U.S.C. § 102(b) over U.S. Patent No. 6,199,126 B1 (Auerbach). Claims 5 and 6 were rejected under 35 U.S.C. § 103(a) over Auerbach in view of U.S. Patent No. 6,691,305 (Henkel). Applicant respectfully traverses the rejections and submits that the claims herein are patentable in light of the arguments below.

The Auerbach Reference

Auerbach is directed to an apparatus and method for transparent on-the-fly decompression of the program instruction stream of a processor. According to Auerbach, a decompression device is connected between a processor and a memory storing compressed information. The decompression device receives a request from

the processor for information, retrieves compressed information from the memory, decompresses the retrieved compressed information to form uncompressed information, and transmits the uncompressed information to the processor. The compressed information may include both program instructions and data. When the decompression device receives a request for information, which includes an unmodified address, from the processor, it generates an index offset from the received unmodified address. An indexed address corresponding to the generated index offset is retrieved from an index table. Compressed information corresponding to the selected indexed address is retrieved from the memory and transmitted to the processor (*Auerbach Abstract*).

The Henkel Reference

Henkel is directed to a method and apparatus for compressing and decompressing object code instructions that are included in a software program that executes on a computer system. (See *Henkel: Col. 1, lines 16-20*). According to Henkel, the method includes extracting compressible instruction and data portions from executable code, creating a mathematical model of the extracted code portions, class the individual instructions in the extracted portions based upon their operation codes and compressing the instructions. (*Henkel Abstract*).

The Claims are Patentable Over the Cited References

The present application is generally directed to micro controller that processes compressed codes stored in a memory.

As defined by the amended independent Claim 1, a micro controller, including a CPU, performs processing in accordance with a program. The micro controller further includes a memory, storing: grouped compressed codes resulting from the conversion of original codes into variable length codes in a plurality of compressed code format

types, and each type has a fixed length; an address conversion information, specifying the head address of each group of grouped compressed codes of variable lengths; and a compressed code type information in blocks corresponding to the groups of the compressed codes. The compressed code type information includes compressed code format type data corresponding to the compressed codes, and indicates the compress code format types of the corresponding compressed codes contained in each group. A compressed code processing part, specifies, from a code address outputted by the CPU, an address conversion information and compressed code type information to be referred, and uses the specified address conversion information and the compressed code type information to determine the corresponding compressed code address and to read the corresponding compressed code.

The applied references do not disclose or suggest the features of the present invention as recited by the claims as amended. In particular, Auerbach and Henkel do not disclose or suggest, (1) "a memory, storing: grouped compressed codes, resulting from the conversion of original codes into variable length codes in a plurality of compressed code format types, and each type has a fixed length," and (2) "a memory, storing: ... a compressed code type information in blocks corresponding to the groups of the compressed codes, including compressed code format type data corresponding to the compressed codes, and indicating the compress code format types of the corresponding compressed codes contained in each group," as recited in the amended Claim 1.

(1) Auerbach does not even teach or suggest compressing the original codes into compressed codes of variable lengths, let alone those compressed codes being in a plurality of format types. Figs. 6a and 6b illustrate the only compression format taught by Auerbach (*also see Auerbach col. 5, lines 23-49*). In Auerbach, each instruction is 32-bit long (*Auerbach Fig. 6b, element 650*). Each address is compressed in to a 16-bit long compressed code (*Auerbach Fig. 6b, element 686; "00FD" is the compressed code*

of the instruction "11112222; "0102" is the compressed code of the instruction 33334444). This is the only code compression scheme taught by Auerbach. Since Auerbach teaches only the one compressed code format type of 16-bit in length, Auerbach cannot teach or suggest the compressed codes belonging to "a plurality of compressed code format types, and each format has a fixed length" recited in Claim 1.

In Fig. 6b of Auerbach, the 3rd instruction in element 650 "21324354" remains uncompressed and 32-bit long. The above features of Claim 1, however, are directed to "compressed codes." The uncompressed instruction, such as "21324354" in Fig. 6b, is thus not a compressed code format type as recited in Claim 1.

The Action asserts Auerbach teaches compressing the instructions into codes of variable lengths by citing Auerbach col. 3, lines 7-8 and col. 6, lines 20-23. Here, the Action equates each instruction of Auerbach as a "code" recited in the claims because the cited portions are directed at storing compressed codes.

The cited portions of Auerbach, however, do not teach or suggest those compressed instructions are of variable lengths as the Action asserts. Auerbach col. 3, lines 7-8 recites only that the "instructions [are] stored in the compressed format;" (Id.) that portion makes no reference to the code length of the compressed instruction. Auerbach col. 6, lines 15-23 recites the following:

In a preferred embodiment, a PowerPC processor is used. The PowerPC instruction set requires a relatively large amount of memory for the instructions which perform a given task. This is because the PowerPC architecture is a RISC architecture and uses a 32-bit, fixed length instruction set. This results in a requirement for a larger amount of memory (ROM, Flash, or DRAM) for instruction storage than a processor which uses either smaller fixed length or variable length instructions.

That portion of Auerbach is not even directed at the compressed instructions! It merely states the PowerPC instruction set employs fixed length 32-bit instructions and, if left uncompressed, would require a larger amount of memory to store than a processor using either "smaller fixed length or variable length instructions." No length of the compressed codes, variable or otherwise, is even discussed in that portion of Auerbach.

Other teachings of Auerbach also fail to teach the features of Claim 1. Fig. 8 of Auerbach illustrates compressing each 64-byte block into compressed block of 16 – 76 bytes in length. That figure, however, does not read on the features of Claim 1. As discussed above, the Action equates a “code” of Claim 1 with an instruction of Auerbach. Fig. 8 of Auerbach illustrates compressing 64-byte or 16 instructions into compressed blocks of variable lengths (*each instruction is 32-bit long, see Auerbach col. 3, lines 7-8*). That figure does not teach or suggest the length of each compressed instruction, and, on the other hand, the “variable length codes” features of Claim 1 are directed the code lengths of the compressed codes or instructions. Accordingly, the teaching Auerbach, Fig. 8 does not teach or suggest feature “grouped compressed codes, resulting from the conversion of original codes into variable length codes in a plurality of compressed code format types, and each type has a fixed length” of Claim 1.

In the case the Office wishes to changes its assertion of “code” recited in the claims, and asserts each 64-byte block as such “code,” Applicant respectfully notes that the compressed blocks range randomly from 16 – 76 bytes in length. The compressed blocks of random lengths do not read on the feature “variable length codes in a plurality of compressed code format types, and each type has a fixed length” of Claim 1.

Alternatively, the Office may wish to assert that compressed codes of variable lengths may be inferred from the compressed blocks of variable lengths. That assertion, however, cannot be made without running afoul of the enablement issue. See MPEP 2121.01, reciting:

“In determining that quantum of prior art disclosure which is necessary to declare an applicant's invention ‘not novel’ or ‘anticipated’ within section 102, the stated test is whether a reference contains an ‘enabling disclosure’... ” *In re Hoeksema*, 399 F.2d 266, 158 USPQ 596 (CCPA 1968). The disclosure in an assertedly anticipating reference must provide an enabling disclosure of the desired subject matter; mere naming or description of the subject matter is insufficient, if it cannot be produced without undue experimentation. *Elen Pharm., Inc. v. Mayo Found. For Med. Educ. & Research*, 346 F.3d 1051, 1054, 68 USPQ2d 1373, 1376 (Fed. Cir. 2003).

As discussed above, Auerbach teaches but one compression scheme resulting in compressed codes of fixed 16-bit in length. Merely showing that compressed blocks may be of variable lengths is insufficient to read on the features of Claim 1.

In contrast, Claim 1 recites that the compressed codes are in a plurality of compressed code format types. Each of the format type has a fixed length. Applicant's specification Figs. 2A – 2D illustrates examples of the compressed code format types. In Fig. 2A, a compressed code of type 0 is 4-bit long; in Fig. 2B, a compressed code is of type 1 is 8-bit long; in Fig. 2C, a compressed code of type 2 is 10-bit long; in Fig. 2D, a compressed code of type 3 is 16-bit long. (See also Applicant's specification, page 9, last paragraph reciting "FIG. 2A, FIG. 2B, FIG. 2C, and FIG. 2D are diagrams for explaining the compressed code formats of a first embodiment that illustrates the basic principles of this invention. Through four types of formats shall be described here as an example,").

Applicant thus respectfully submits that the Action clearly erred in asserting that Auerbach teaches compressed, "variable length codes" recited in Claim 1. Auerbach does not even teach or suggest the "variable length codes" feature, let alone "...variable length codes in a plurality of compressed code format types, and each type has a fixed length," as recited in Claim 1.

(2) In case the Office insists on the position of Auerbach teaching compressed codes of "variable length codes in a plurality of compressed code format types, and each type has a fixed length" recited in Claim 1, Auerbach still fails to teach or suggest the features relating to the "compressed code type information." The Action suggested the Auerbach's block addressing scheme (see *Auerbach col. 8, line 61 – col. 10, line 2*) as reading on the "compressed code type information" features. Applicant respectfully traverses.

First, Auerbach's addressing scheme relied upon by the Action is directed to addressing of the 64-byte blocks, and not that of instructions. (See *Auerbach col. 8, line*

62 – col. 9, line 10; Fig. 9). Auerbach teaches an index table (Auerbach, Fig. 9) having a list of 32-bit wide entries, with each entry indicating the address of two sequential blocks of variable length compressed instruction streams. Each of the blocks is 64-byte long. The index table 910 includes a 26-bit address field 912 and a 6-bit offset field 914. The 26-bit address field 912 is the address of the first of the two compression blocks in the compression region. The 6-bit offset field 914 is an offset from that address to the next compressed block. The 6-bit offset field 914 is in a format that locates the second block of code and indicates whether those blocks have been compressed or not, as shown in TABLE 1 Auerbach of . Id.

Since the Action's position on the "variable length codes" features relies on asserting the instruction of Auerbach as the "code" of Claim 1, Auerbach's block addressing scheme cannot read on features relating to "compressed code type information" of Claim 1, which are directed to the addressing of the compressed codes (or compressed instructions per the Action's assertions). In particular, Claim 1 recites "...compressed code format type data corresponding to the compressed codes, and indicating the compress code format types of the corresponding compressed codes contained in each group." Since the cited addressing scheme of Auerbach is directed to block addresses only, Auerbach cannot teach or suggest having a compressed code type format datum corresponding to one of the compressed instructions, or that the compressed code type format datum indicates the compress code format type of the corresponding compressed instruction.

Second, even if the Office insists on applying Auerbach's block addressing scheme to the features of Claim 1, Auerbach still fails to teach or suggest "compressed code format type data ... indicating the compress code format types of the corresponding compressed codes contained in each group." Auerbach's block addressing scheme is directed at a two-block addressing. In particular, the 26-bit address field 912 stores the address of the first of the two compression blocks in the

compression region. The 6-bit offset field **914** is an offset from that address to the next compressed block. The 6-bit offset field **914** is in a format that locates the second block of code and indicates whether those blocks have been compressed or not, as shown in TABLE 1. *Id.* (See *Auerbach col. 8, line 62 – col. 9, line 10; Fig. 9*). In particular, Auerbach's TABLE 1 indicates the following:

6-bit offset value	Representation
0	Both 64-byte blocks are uncompressed
1	1 st block is uncompressed; 2 nd block is compressed
>1	The length of the compressed 1 st block

As shown above, the 6-bit offset **914** carries information on (a) the length of compressed 1st block and (b) the compression statue of the 1st and 2nd blocks. Neither (a) the length of compressed 1st block nor (b) the compression statue of the 1st and 2nd blocks can be said to be "compressed code format type" of Claim 1. The information (a) the length of compressed 1st block is plainly not a format type. The information (b) informs whether the blocks are compressed, but no the types of the compressed codes. As discussed above, being not compressed is not a "compressed code format type. The 6-bit offset **914** thus cannot read on the "compressed code format type data," which indicates "the compress code format types of the corresponding compressed codes contained in each group."

In contrast, Fig. 6 of Applicant's specification illustrates a block of the compressed code type information including 16 entries of compressed code format type data, each datum corresponds to a compressed code (*elements a, b, c, ...p, corresponding to the compressed codes a-p in Fig. 3B of Applicant's specification*), and indicates the format types of the corresponding compressed codes ("00" indicates type 0; "01" indicates type 1; "10" indicates type 2; "11" indicates type 3). Since the

compressed code length is known (as illustrated in Figs. 2A-2D and discussed above), Applicant's system can calculate address of each compressed code. The advantages of Applicant's system is not seen to flow from the teaching of Auerbach. Auerbach, in fact, acknowledges the weakness of its invention, which Applicant's invention addresses. (See *Auerbach*, col. 7, line 65 – col. 8, line 5, reciting *"The worst latencies will occur when the CPU executes a jump out of the current 64-byte block. If the jump is to a location which is not the beginning of a 64-byte block, the decompressor must begin fetching and decoding serially until it reaches the desired instruction and can return them to the CPU."*).

In sum, Auerbach does not teach or suggest the features of Claim 1, and the withdrawal of the rejections based on Auerbach is respectfully requested. Applicant respectfully summarizes Applicant's positions below:

(1) Auerbach teaches only type of compression format where a 32-bit instruction is compressed into a 16-bit compressed code, or is left uncompressed. (See *Auerbach* col. 5, lines 23-49; Fig. 6). Auerbach thus cannot teach or suggest compressed original codes (instructions) in "variable length codes."

(2) The Action equates the instruction of Auerbach as the "code" in Claim 1, and cites Auerbach col. 3, lines 7-8 and col. 6, lines 20-23 as teaching the compressed "variable length codes" feature.

(3) The Action clearly erred in making that assertion. Auerbach col. 3, lines 7-8 merely discloses that the instructions are stored in compressed form. No length of the compressed instructions is discussed or disclosed.

(4) Auerbach col. 6, lines 20-23 discloses that the PowerPC processor uses fixed 32-bit instructions. That instruction set requires a larger memory to store than a processor using a shorter, fixed-length instructions, or a processor using variable-length instructions. Again, no length of the compressed instructions is discussed or disclosed, let alone the "variable length codes" feature.

(5) The Action improperly conflates a compressed block with a compressed code (the instruction in Auerbach, as asserted in the Action). The Action cites Auerbach's block addressing scheme against features relating to "compressed code type information" of Claim 1. However, those features of Claim 1 are directed at compressed code addressing, and Auerbach's block addressing scheme cannot apply.

(6) The Action further asserts Auerbach's 6-bit offset 914 as reading on the a compressed code type information storing compressed code format type data. However, the 6-bit offset 914 indicates only (a) the length of compressed 1st block and (b) the compression statue of the 1st and 2nd blocks. Neither (a) nor (b) can be said to be compressed code information type.

(7) Applicant further notes that the advantages of present invention are not seen to flow from Auerbach's teaching. For example, as discussed above, Auerbach acknowledges that its scheme cannot decode the address of each compressed code, and needs to uncompressed the codes serially to reach the requested compressed code (See *Auerbach*, col. 7, line 65 – col. 8, line 5). In contrast, Applicant's invention includes a compressed code type information storing compressed code format type data, which correspond to the compressed the codes, and indicate the compressed code format types of the compressed the codes. Each compressed code format types has a fixed length. Accordingly, a system according to Applicant's invention can calculate the address of each of the compressed codes by adding the lengths of the compressed code format types.

Accordingly, Auerbach does not disclose or suggest those features of amended independent Claim 1. The ancillary Henkel reference is not seen to seen to remedy the deficiencies of Auerbach.

Since the applied references, alone or in combination, do not disclose or suggest the features of the present invention as recited by the amended independent Claim 1,

those references cannot be said to anticipate nor render obvious the invention which is the subject matter of that claim.

Accordingly, independent Claim 1 as amended is believed to be in condition for allowance and such allowance is respectfully requested.

New Claim 10 recites similar features as Claim 1, and is thus allowable as per Claim 1. Moreover, Claim 10 recites "a compressed code processing part, determining, from a code address output by the CPU and via which the code address the CPU specifies one of the original codes ..., a compressed code address of a compress code corresponding to the specified original code," which is not taught by Auerbach. First, as discussed, Auerbach does not teach or suggest addressing individual compressed code (*or instruction per the Action's position; see Auerbach, col. 7, line 65 – col. 8, line 5*); instead, Auerbach teaches only decoding for the address of a 64-byte block. Auerbach thus fails to teach or suggest all the features of Claim 10. The allowance of Claim 10 is respectfully requested.

The remaining claims depend either directly or indirectly from amended independent Claim 1 and recite additional features of the invention which are neither disclosed nor fairly suggested by the applied references, and are also believed to be in condition for allowance. As such, reconsideration and allowance of those claims are respectfully requested.

Conclusion

In view of the foregoing, it is respectfully submitted that the application is in condition for allowance. Reexamination and reconsideration of the application, as amended, are requested.

If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is requested to call the undersigned attorney at the Los

Appl. No. 10/611,315
Amdt. Dated December 18, 2008
Reply to Office Action of July 28, 2008

Attorney Docket No. 83394.0008
Customer No.: 26021

Angeles, California telephone number (310) 785-4721 to discuss the steps necessary for placing the application in condition for allowance.

If there are any fees due in connection with the filing of this response, please charge the fees to our Deposit Account No. 50-1314.

Respectfully submitted,
HOGAN & HARTSON L.L.P.

Date: December 18, 2008

By: 

Dariush G. Adji
Registration No. 51,386
Attorney for Applicant(s)

1999 Avenue of the Stars
Suite 1400
Los Angeles, CA 90067
Phone: (310) 785-4600
Fax: (310) 785-4601